POSE: Protobuf Signing and Encryption for Location Proof Systems on Constrained Devices 

Miguel Francisco[0000−0003−1759−3167], Samih Eisa[0000−0003−0972−4171], and Miguel L. Pardal[0000−0003−2872−7300]

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal
{miguel.c.francisco,miguel.pardal}@tecnico.ulisboa.pt
samih.eisa@inesc-id.pt

Abstract. Existing location proof systems provide reliable and verifiable information about the location of user devices, usually smartphones. The proofs require short-range communication with other devices, called witnesses, that endorse and strengthen location claims. However, extending these proof systems to constrained devices in Internet of Things (IoT) applications has been a source of security challenges. A specific problem is the use of Bluetooth Low Energy (BLE). BLE is a widely used, low-power variation of the classic Bluetooth. Although it has matured over the years, it still presents vulnerabilities that can compromise the authenticity of the packets exchanged between the devices of witnesses. In this paper, we present POSE, an end-to-end application layer security protocol for BLE. POSE uses protocol buffers for message serialization/deserialization and can provide message confidentiality and authenticity. We implemented POSE and tested it with an application that issues medical appointment attendance proofs.

Keywords: Location Proof System · Internet of Things · Constrained Devices · Security · Encryption · Bluetooth Low Energy · Protocol Buffer.

1 Introduction

Object security has been successfully used in application-layer solutions in the IoT world [9], without depending on the message-format encoding. JOSE [RFC 7515, 7516] and COSE [RFC 8152] make use of this concept to guarantee object security using JSON and CBOR format-encoding messages, respectively. However, IoT applications face significant resource constraints and may benefit from additional efficiency improvements. One possible improvement over the existing solutions is the use of Protocol Buffers [6] (usually shortened to just *Protobuf*). These buffers are an efficient and lightweight way of encoding data, while also providing language and platform-independent data abstractions. Although this encoding mechanism has been around for many years, it has never been properly explored for object security when used in BLE communications. Such short-range communications are used in different Location Proof Systems, where location proof techniques require BLE communications between provers and witnesses to exchange location claims and endorsements.

Location Proof Systems [12,1] are essential enablers for secure and reliable Location-Based Services (LBS) that protect information about the location of users on smart devices. The functionalities of these systems can be leveraged with the usage of IoT and constrained devices, innovating the way to seamlessly produce and verify location proofs.

In this paper, we present POSE, a proposal to secure the exchange of messages between constrained devices that use BLE for communication. The solution offers Protobuf-based object security in BLE messages exchanged between devices. We demonstrate the use of POSE in an application supported by a location proof system, with devices operating as provers and witnesses.

The remaining paper has the following contents: the POSE specification can be seen in Section 3; in Section 4 we show how to implement it POSE and a use case specific proof-of-concept, just before the conclusion in Section 5.

2 Related Work

Most of the location certification systems are inspired by the notion of Location Proof given by Saroiu et al. [7]. The idea of having the users of the system working as witnesses for other users and certifying their location claims is also a method used by multiple systems. In APPLAUS [12] and CREPUSCOLO [1] the communication between both roles is made in a direct way, meaning that there is no intermediary in the communication, making it faster. Both systems make efforts to minimize collusion attacks but cannot eliminate the attack in all cases. This is true even in more recent systems, such as PASPORT [5]. Recently, there has been an effort to provide better tools for using location proofs, such as the SureThing framework [2] In all of these systems, it is essential to have proximity communication with witnesses and, on top of that, to have secure communication with them, including authentication and integrity protection.

Bluetooth has been one of the chosen protocols to realize such short-range communications in the past years [10,12], as well as BLE for constrained devices. In BLE, if both connecting devices do not force the pairing protocol, the communication is done in plain-text but even if they do, they are still vulnerable to some design flaws like Man-In-The-Middle (MITM) attacks and downgrade attacks of the communication to plain-text [11]. Therefore, confidentiality is important to protect sensitive data in those message exchanges. Tjäder [8] showed how to guarantee message confidentiality in those exchanges by enhancing those communications with COSE [RFC 8152]. CBOR Object Signing and Encryption (COSE) is an IETF standard on how to process signatures, encryption, and Message Authentication Codes (MAC) computations for CBOR (Concise Binary Object Representation) message-encoding format.

Protocol buffers (Protobuf) have already been introduced in the world of heterogeneous constrained devices as a lightweight and interoperable alternative to standardized message encoding schemes like CBOR, JSON or BSON [6]. Jenkov [3] has shown in practice that Protobuf messages with five different fields have a much higher read and write throughput than equivalent CBOR messages.

3 POSE

The POSE (**P**rotocol buffer **O**bject **S**igning and **E**ncryption) protocol is designed to provide an end-to-end security layer over BLE communication on constrained devices. The protocol is based on the COSE (CBOR Object Signing and Encryption) specification [RFC 8152] which is created to provide basic security services for the CBOR (Concise Binary Object Representation) data format [RFC 8949]. The COSE specification describes in detail how to create and process message signature, authentication codes and encryption using CBOR as the data encoding format for small size messages on constrained and limited devices. Fundamentally, POSE follows the standard COSE specification. However, it takes full advantage of Protobuf as its underlying data encoding format to abstract the exchanged messages over BLE communications. Protobuf provides a convenient way to structure data in a compact way and then use a Protoc, a protocol buffers code generator, to easily write and read those structured data [4].

3.1 Message Format and Types

POSE specifies different message types depending on the security guarantee to ensure and the number of recipients or senders. In the context of a single recipient one-way BLE communications, POSE specifies three message types, each offering different security guarantees, detailed in Section 3.1. All of them follow the same grammar and present the same fields as all the remaining COSE objects.

Message Format POSE protocol supports the same primitive types as defined in COSE specification, like Booleans, Integers, Byte strings, etc. Each field in a POSE object message can be a primitive type or Protobuf-defined message, depending on the type of the message, however, it always starts with the three fields: protected header parameters, unprotected header parameters and the content of the message. Both protected and unprotected headers are 'label'-'value' maps. Each label is a well defined Integer, while a value is a primitive type or another POSE message. The protected bucket contains parameters about the current layer and information about the used cryptographic algorithms. All this information is protected since it is used in the cryptographic computation, where is either signed, hashed or used as associated data in the encryption, as explained in Section 4.1. The unprotected field is similar but not protected, meaning that it may not be authentic when it reaches the recipient. Both fields are serialized into a byte string. The content of the message field can be either plain-text, cipher-text or another POSE message.

Message Types The different POSE message types can be seen in Table 3.1. These messages depend on the knowledge of who is the recipient and on an implicit symmetric key/asymmetric key pair, previously established.

Table 1. POSE Message Types.

POSE Message Type	Semantics
POSE_Sign1	Signed data object
POSE_Mac0	Mac data object
POSE_Encrypt0	Encrypted data object

```
message POSE_Sign1{
    bytes protected = 1;
    HeaderMap unprotected = 2;
    bytes payload = 3;
    bytes signature = 4;
}
```

Listing 1: POSE_Sign1 Protocol Buffer definition.

Signed data object The POSE_Sign1 message type object ensures the non-repudiation of the transmitted data, by one signer only. It makes use of the same signature algorithms as used in the COSE standard, which are the Edwards-curve Digital Signature Algorithm (EdDSA) and the Elliptic Curve Digital Signature Algorithm (ECDSA). The protected bucket and payload are both signed, where the latter can even include another POSE message type, allowing the creation of sealed objects. The POSE_Sign1 protocol buffer definition can be seen in Listing 1.

Mac data object The POSE_Mac0 message type object ensures integrity and authenticity of both the payload and the protected bucket by generating a tag with a Message Authentication Code (MAC). It makes use of the same algorithms as COSE, which can either be a block cipher algorithm, like AES-MAC, or a hash algorithm, like HMAC. The Protobuf definition of the POSE_Mac0 message type can be seen in Listing 2.

```
message POSE_Mac0{
    bytes protected = 1;
    HeaderMap unprotected = 2;
    bytes payload = 3;
    bytes tag = 4;
}
```

Listing 2: POSE_Mac0 Protocol Buffer definition.

```
message POSE_Encrypt0{
    bytes protected = 1;
    HeaderMap unprotected = 2;
    bytes ciphertext = 3;
}
```

Listing 3: POSE_Encrypt0 Protocol Buffer definition.

Encrypted data object The POSE_Encrypt0 message type object is the one that offers the most security guarantees and is the type in which we focused on in our implementation (Section 4). This type ensures message confidentiality and integrity and message freshness. The Protobuf definition of the POSE_Encrypt0 message type can be seen in Listing 3.

3.2 Message Confidentiality and Integrity

To ensure both the confidentiality and integrity of the transmitted data, the POSE_Encrypt0 message must be used, instead of the POSE_Mac0 object which only guarantees integrity. It uses the Authenticated Encryption with Associated Data (AEAD) as the form of encryption and the AES-GCM as the mode of operation for the encryption computation of the payload. AEAD ensures the authenticity of the associated data used in encryption along with the integrity and confidentiality of the cipher-text. The encryption makes use of a pre-shared 128-bit symmetric key. We leave a possible handshake for a key agreement for future work. This mode of operation requires an Initialization Vector (IV) to guarantee the uniqueness of each encryption, which we randomize for each message and use to guarantee the message freshness as well, as shown in Section 3.3.

3.3 Message Freshness

Although the AEAD with AES-GCM offers a lot of security guarantees, it does not guarantee the freshness of the transmitted data. This problem is not addressed in the COSE standard. POSE, however, solves this problem by using the IV of the AEAD encryption as a nonce field.

4 Implementation

This Section presents the implementation of the POSE protocol. In Section 4.1 we detail how to build the POSE_Encrypt0 message object according to format and fields previously explained. In Section 4.2 we show how POSE is used in an example application.

```
message Enc_Structure{
    string context = 1;
    bytes protected = 2;
    HeaderMap unprotected = 3;
    POSE_Encrypt0 body = 4;
}
```

Listing 4: Enc_Structure Protocol Buffer definition.

4.1 Security processing

The first step to produce a POSE_Encrypt0 message object is to create a consistent byte stream for the associated data that will be used in the cryptographic computation. For that purpose, a new object Enc_Structure should be created according to the Protobuf definition shown in Listing 4, as defined by the POSE protocol. The subsequent steps describe the encryption and decryption processes.

Encryption

1. Create the Enc_Structure message object and fill it with the appropriate fields such as the context string and the protected attributes. The context string is a well-defined string that identifies the type of the payload.
2. Serialize the created Enc_Structure object into byte stream using protocol buffers encoding to create the associated data.
3. Call the encryption algorithm with the previously loaded encryption key K, the plain-text of the POSE_Encrypt0 message object, and the associated data (AD). Then include the result in the cipher-text field of the POSE_Encrypt0.

Decryption

1. Create an object of Enc_Structure message similar to the one received but without the cipher-text.
2. Serialize the created Enc_Structure object into byte stream using protocol buffers encoding to create the associated data.
3. Call the decryption algorithm with the previously loaded decryption key K, the cipher-text of the POSE_Encrypt0 object from the received Enc_Structure message, and the associated data (AD).

4.2 Example application

The example application allows a patient to prove his presence at a medical clinic. This was achieved through one-way BLE communication that allows the patient,

```
{
    context: "Encrypt0"
    protected: map: {5:
                            E09057384768B9557658E309} //IV/Nonce
    unprotected : {}
    body {  //POSE_Encrypt0
        protected: map: {1: 10}
        unprotected {}
        ciphertext: 1FDF435D9C2C80B3C457FB18E85B5C93DA33A9BC...
    }
}
```

Listing 5: Enc_Structure implementation.

using a running app on his smartwatch, to exchange messages with a Kiosk device at the medical clinic to prove his presence. The patient acts as a location prover and the Kiosk as a witness to endorse the patient's presence at the clinic. The security needs to be guaranteed by successfully applying the POSE protocol to provide the desire BLE application security level. The exchanged BLE packet between the smartwatch and the Kiosk device uses our implementation of the Enc_Structure, which can be seen in Listing 5.

The cipher-text is an encrypted location claim. The patient generates a location claim with his app, containing the user, location, and time information. Then the patient signs the claim to guarantee the non-repudiation of his claim. This discards the necessity to use the POSE_Sign1 message object, previously explained. The smartwatch advertises BLE packet with the Enc_Structure object to the Kiosk device. Without POSE, the signed location claim would be in plain-text and, therefore, an attacker could read the user sensitive information. However, with POSE implemented, we guarantee the authenticity and confidentiality of the patient's location claim. The Kiosk device on the other end follows the decryption methodology, previously explained, and signs the location claim creating a location endorsement to be submitted to a location service provider as part of the location proof system. The implemented process of this medical appointment use case can be seen in Figure 1.

5 Conclusion

In this paper we presented POSE, an application-layer security enhancement protocol for Bluetooth Low Energy (BLE) communications on constrained devices, using Protobuf as data-encoding format. POSE was tested with an example application supported by a location proof system that allows users to prove their locations and presence with their smart devices. The results show the feasibility and efficiency of POSE protocol on top of pairing-less BLE connections with

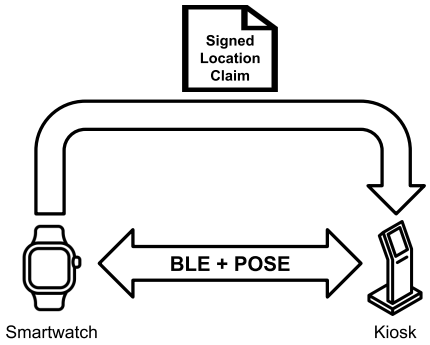

Fig. 1. Location proof technique used to test the POSE implementation.

small packet overhead and small increase in CPU usage. This same approach and implementation shows great promise for many other IoT applications.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020 (INESC-ID) and through project with reference PTDC/CCI-COM/31440/2017 (SureThing).

References

1. Canlar, E.S., Conti, M., Crispo, B., Di Pietro, R.: Crepuscolo: A collusion resistant privacy preserving location verification system. In: 2013 International Conference on Risks and Security of Internet and Systems (CRiSIS). pp. 1–9 (Oct 2013). https://doi.org/10.1109/CRiSIS.2013.6766357
2. Ferreira, J., Pardal, M.L.: Witness-based location proofs for mobile devices. In: 17th IEEE International Symposium on Network Computing and Applications (NCA) (Nov 2018)
3. Jenkov, J.: Rion vs. json vs. protobuf vs. messagepack vs. cbor (Sep 2019), http://tutorials.jenkov.com/rion/rion-performance-benchmarks.html
4. Kaur, G., Fuad, M.M.: An evaluation of protocol buffer. In: Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon). pp. 459–462 (March 2010). https://doi.org/10.1109/SECON.2010.5453828
5. Nosouhi, M.R., Sood, K., Yu, S., Grobler, M., Zhang, J.: Pasport: A secure and private location proof generation and verification framework. IEEE Transactions on Computational Social Systems **7**(2), 293–307 (April 2020). https://doi.org/10.1109/TCSS.2019.2960534
6. Popic, S., Pezer, D., Mrazovac, B., Teslic, N.: Performance evaluation of using protocol buffers in the internet of things communication. In: 2016 International Conference on Smart Systems and Technologies (SST). IEEE (oct 2016). https://doi.org/10.1109/sst.2016.7765670

7. Saroiu, S., Wolman, A.: Enabling new mobile applications with location proofs. In: Proceedings of the 10th Workshop on Mobile Computing Systems and Applications. HotMobile '09, Association for Computing Machinery, New York, NY, USA (2009), https://doi.org/10.1145/1514411.1514414
8. Tjäder, H.: End-to-end Security Enhancement of an IoT Platform Using Object Security. Master's thesis, Linköping University, Information Coding (2017)
9. Vučinić, M., Tourancheau, B., Rousseau, F., Duda, A., Damon, L., Guizzetti, R.: Oscar: Object security architecture for the internet of things. In: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014. pp. 1–10 (2014). https://doi.org/10.1109/WoWMoM.2014.6918975
10. Wang, X., Pande, A., Zhu, J., Mohapatra, P.: STAMP: Enabling privacy-preserving location proofs for mobile users. IEEE/ACM Transactions on Networking **24**(6), 3276–3289 (dec 2016). https://doi.org/10.1109/tnet.2016.2515119
11. Zhang, Y., Weng, J., Dey, R., Jin, Y., Lin, Z., Fu, X.: On the (in)security of bluetooth low energy one-way secure connections only mode. ArXiv **abs/1908.10497** (2019)
12. Zhu, Z., Cao, G.: Applaus: A privacy-preserving location proof updating system for location-based services. In: Proceedings IEEE INFOCOM. pp. 1889–1897 (April 2011). https://doi.org/10.1109/INFCOM.2011.5934991